Machine Learning Project - Image Classification

Luis Marcos Lopez Casines
luismarcos.lopezcasines@studenti.unipd.it

Laia Porcar Guillamon
laia.porcarguillamon@studenti.unipd.it

1. Introduction

Images are a big and important part of all the data that is produced in the world on a daily basis. Image classification is the task of categorizing and assigning labels to groups of pixels or vectors within an image following particular rules. Image classification is used in many areas, such as quality management, animal monitoring, medical imaging, object identification in satellite images, brake light detection, machine vision, etc. In this project we are going to classify images of clothing items. This can be useful for tasks as stocking items in an online store and buyer behavior analysis.

The main goal is to find the best machine learning algorithm to classify the data, among eight commonly used classification algorithms: Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), XGBoost (XGB) [1], Ridge Classifier (RC) and Neural Networks (NN) [2].

Finding the best algorithm means finding out which of them yields the highest accuracy score on the validation data. The best performing algorithm is then tested on the test set to obtain the accuracy at which we are able to classify new clothing images. Through the study of the different algorithms and different combinations of hyperparameters within them, we have concluded that the best performing one is SVM, although closely followed by XGB. The accuracy on the test set for this algorithm is 0.8965. The confusion matrix shows that the class it has more problems with is the sixth.

2. Dataset

The dataset we are working with in this project is the Fashion-MNIST [6], a dataset of Zalando's article images. It has been taken from the Kaggle competition: ML Project - Image Classification. It consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 gray-scale image, associated with a label from 10 classes. The classes are the following: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. However they are identified with numbers from 0 to 9 in our dataset.

Since we are already given a test set we separate the training data into training and validation. For the validation set we take 12,000 examples, which represents a 20% of the original training set. We check for unbalance in the data but find out that the training and validation sets are mostly well balanced. The biggest difference of examples between two classes is 87, which corresponds to less than 2% of the average number of examples in a class. The provided test set is perfectly balanced with 1000 examples per class.

Next, we proceed to scale the data with the MinMaxScaler from sklearn [4]. This is a good way to assure the algorithms we'll be using will behave correctly. Some of them, for instance LR, SVM and KNN, are more sensible to the scaling of the data.

The main issue with our data is the big amount of features we have (784 after flattening our 28 x 28 images), which together with the non negligible 60,000 examples, makes some algorithms take a considerable amount of time to be trained. This problem becomes more worrisome when we want to implement a GridSearchCV to look for the best performing parameters for each of the algorithms. We have decided to tackle this issue by somehow reducing the impact of the 784 features in the training of our models. And this in turn has been done in two ways: by dimensionality reduction using Principal Component Analysis (PCA) and by discarding the less important features using Univariate Feature Selection (UFS). The amount of data being used to find the optimal hyperparameters is 20,000 examples, slightly more than 40% of the total.

PCA is used to reduce the data dimensionality by projecting it to a lower dimension environment. It attempts to find out what features explain the most variance in the data [5]. The important question is therefore how many dimensions to reduced the data to. We need a number of dimensions that eases our process but at the same time keeps the essence of the data. To help us find out an answer we plot the cumulative explained variance against the number of components (Fig. (1)). A dimension of 100 seems to be a compromise between both.

UFS on the other hand, works by selecting the best features based on univariate statistical tests, in our case a Chi-Square test (χ^2). In feature selection, we aim to select the

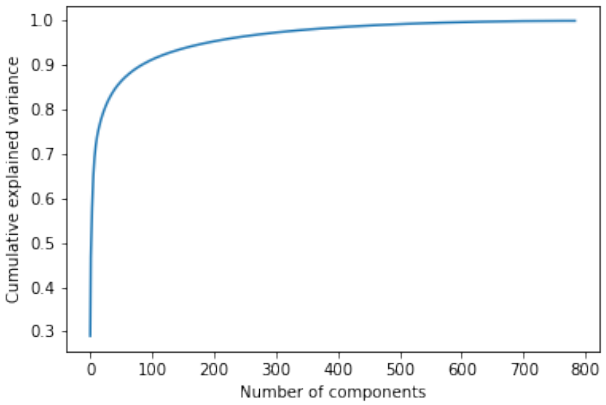

Figure 1: Cumulative explained variance vs number of components. The curve quantifies how much of the total, 784-dimensional variance is contained within the first N components.

features which are highly dependent on the response (the target variable). The higher the Chi-Square value, the more response dependent the feature is and thus it can be selected for model training [3]. We select the most important 392 features (50% of the total).

Following two different approaches allows more assurance that the data is not being underrepresented too much, while making the process of finding the best hyperparameters with the GridSearchCV more doable time wise. We can run each in under 30 minutes.

3. Method

As said before, the goal of this project is to find the best model to classify the given image dataset. To accomplish this we will train eight different algorithms, already mentioned in Section 1.

After doing the pertinent prepossessing of the data, we are interested in choosing the correct hyperparameters to train each of our algorithms, except for the NN which will be addressed later. To do so, we use GridSearchCV and different combinations of commonly used hyperparameters. As stated in Section 2, this would require a considerable amount of time. That's why we decide on finding the best hyperparameters with datasets having the fewest features, while maintaining a good accuracy.

Carrying out a GridSearchCV on both the dataset with reduced dimensionality and the dataset considering only the most important features, we find out the best performing hyperparameters for each of the datasets and keep only the ones with higher accuracy between the two. These selected hyperparameters are then used to train a model for each of the algorithms, but this time on the full original dataset (48,000 scaled examples for the training set). The validation set is then used to calculate predictions and accuracy scores for each model.

A NN is also trained. A Sequential model is used and different number of hidden layers are tried. The input and hidden layers use 'relu' as activation and the output layer implements a 'softmax' activation. The number of units is the same as the number of features for the input layer, half this value for the hidden, and equal to the number of classes for the output layer. An early stopping mechanism with a patience of 3 is used. Once the best number of layers is obtained the corresponding predictions with the validation set are made and the results are compared with those previously obtained for the other models.

Once the best model is identified, we choose it to perform the predictions on the test set. Finally, a classification report is presented as well as the confusion matrix.

4. Experiments

The search for the best hyperparameters for each model is done for the two built datasets by decreasing the number of dimensions and features. The best accuracy scores found for each model and the two datasets are presented in Table 1. Most of the time, both approaches agree on which hyperparameters to choose for each model (refer to the code for proof), and their accuracy scores are very similar. However, the dimensionality reduction analysis proves to be more useful, since the training of this data is faster, yet yielding similar accuracy scores to those of the data treated with UFS.

Model	Accuracy	
	PCA	UFS
Decision Tree	0.7651	0.7805
Random Forest	0.8488	0.857
SVM	0.8865	0.8772
Logistic Regression	0.8439	0.8277
K-Nearest Neighbors	0.8494	0.8410
XGBoost	0.8649	0.8743
Ridge Classifier	0.7999	0.7883

Table 1: Accuracy of the best performing combination of hyperparameters for the dataset of reduced dimensions (PCA) and for the dataset of most important features (UFS), for each model.

Chosen the best performing hyperparameter, all models are trained with the full training set and the accuracy score of the validation set is calculated for each, yielding the results presented in Table 2.

The NN has the same performance with zero and one hidden layers, so we choose the simplest architecture. A summarized history of the accuracy and loss through the epochs is presented in Fig. (2). As we would expect from the plots in Fig. (2), the accuracy tends to increase for both training and validation sets. The loss on the other hand

tends to decrease. The observed behavior suggest the model is correctly trained, not incurring in overfitting. Improvements could be achieved changing the types of layers and activation of these.

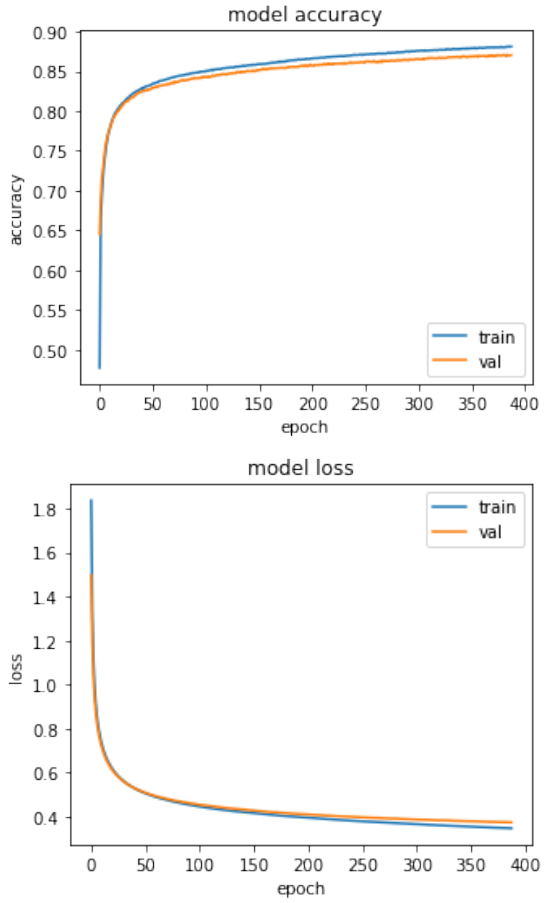

Figure 2: Accuracy (top) and loss (bottom) of the best performing NN model by epochs.

Based on the validation accuracy scores of the eight models, we see that SVM presents the highest result, though closely followed by XGBoost (Table 2). Therefore we calculate the test accuracy and f1 scores for the SVM model, obtaining 0.8965 and 0.8962 respectively.

Finally we visualize the results in detail with a classification report and the confusion matrix. These are displayed in Table 3 and Fig. (3) respectively. Class sixth (shirts) shows the lowest precision. As seen in the confusion matrix, a non-negligible number of class sixth' examples are missclassified as class zero, two and four. This is not very surprising given that these classes are t-shirt/top, pullover and coat respectively.

Model	Accuracy		f1	
	Train	Validation	Train	Validation
DT	0.8476	0.8133	0.8456	0.8098
RF	1.0000	0.8838	1.0000	0.8816
SVM	0.9822	0.9012	0.9822	0.9008
LR	0.8778	0.8563	0.8773	0.8553
KNN	1.0000	0.8604	1.0000	0.8597
XGBoost	1.0000	0.8999	1.0000	0.8997
RC	0.8318	0.8235	0.8288	0.8194
NN	0.8807	0.8704	0.8804	0.8696

Table 2: Training and validation sets accuracy and f1 scores for each model. SVM presents the highest accuracy for the validation set, followed by XGBoost.

Class	Precision	Recall	f1-score	Support
0	0.83	0.84	0.83	1000
1	0.99	0.97	0.98	1000
2	0.81	0.84	0.82	1000
3	0.89	0.90	0.90	1000
4	0.82	0.83	0.83	1000
5	0.97	0.97	0.97	1000
6	0.76	0.70	0.73	1000
7	0.95	0.96	0.96	1000
8	0.97	0.97	0.97	1000
9	0.97	0.96	0.97	1000
Accuracy			0.90	10,000

Table 3: Classification report. Class 6 presents the worse metrics.

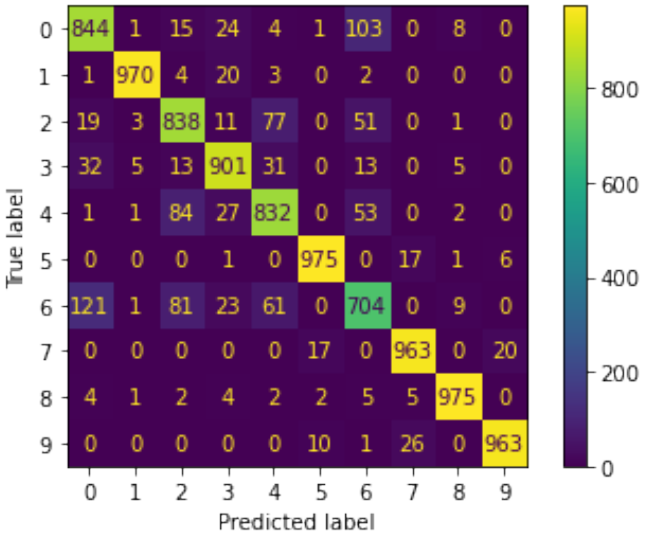

Figure 3: Confusion matrix for the test predictions of the SVM model.

References

[1] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[2] Francois Chollet et al. Keras, 2015.

[3] Sampath Kumar Gajawada. Chi-square test for feature selection in machine learning, 2019. https://towardsdatascience.com/chi-square.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.", 2016.

[6] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.